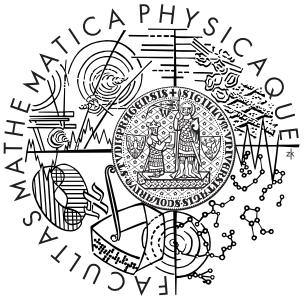
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Peter Majer

Tvorba programu pro zpracování dat tepelné kapacity

Katedra fyziky kondenzovaných látek

Vedoucí bakalářské práce: doc. Mgr. Pavel Javorský, Dr.

Studijní program: Informatika, obecná informatika

2010

Moje poďakovanie patrí predovšetkým vedúcemu bakalárskej práce za jeho trpezlivosť a rady.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 28.5.2010 Peter Majer

Obsah

Název práce: Tvorba programu pro zpracování dat tepelné kapacity
Autor: Peter Majer
Katedra (ústav): Katedra fyziky kondenzovaných látek
Vedoucí bakalářské práce: doc. Mgr. Pavel Javorský, Dr.
e-mail vedoucího: javor@mag.mff.cuni.cz

Abstrakt: Práca popisuje návrh a riešenie programu určeného na spracovanie dát, získaných meraním tepelnej kapacity skúmaných vzoriek. Program má dve časti: spracovanie dát a vykresľovanie grafov. Časť spracúvajúca dáta je vzhľadovo podobná tabuľkovému procesoru - každý stĺpec v tabuľke reprezentuje jedno meranie. Zo stĺpcov reprezentujúcich meranie teplôt je možné spočítať tepelnú kapacitu podľa niekoľkých modelov (uvažujúcich: merné teplo vodivostných elektrónov, fonónový príspevok, Schottkyho príspevok), príp. zo základných matematických operácií zostaviť vlastný výpočet. Spätne je možné fitovať namerané tepelné kapacity na zvolený model. Na nelineárne fitovanie sa používajú simplexová a Marquardtova metóda. Súčasťou programu je aj časť počítajúca Hamiltonián využiteľný pri výpočte energetických hladín. Program je napísaný v jazyku C#.

Klíčová slova: tepelná kapacita, nelineárne fitovanie, C#

Title: Generation of computer program for specific-heat analysis
Author: Peter Majer
Department: Department of Condensed Matter Physics
Supervisor: doc. Mgr. Pavel Javorský, Dr.
Supervisor's e-mail address: javor@mag.mff.cuni.cz

Abstract: This work is about to design and implement the specific-heat analysis application. The application consists of two parts: the first one provides for data manipulation, the second one plots a graph. The data manipulation part resembles a spreadsheet program - each column in a dataset stands for one measurement. Columns containing a temperature data could be used for a specific-heat calculation. This calculation is based on this model functions: conduction electrons consideration, phonon contribution and Schottky contribution. It is also possible to arrange your own formulas by using basic mathematical operations (namely summation, substraction, multiplication and division). Fitting data to the selected model is also possible. Levenberg-Marquardt and simplex method are being used for it. Hamiltonian matrix used for energy levels calculations of magnetic ions in crystalline electric fields could also be calculated. The application is written in C# programming language.

Keywords: specific heat, non-linear fitting, C#

Kapitola 1

Úvod

Cieľom tejto bakalárskej práce bolo vytvoriť použiteľný počítačový program umožňujúci analyzovať dáta experimentov zameraných na skúmanie merného tepla vzoriek.

Základné požadované vlastnosti programu boli

- schopnosť načítať dáta z jednoduchých textových súborov, ktoré sú výstupom experimentálnych aparatúr
- možnosť spravovať dáta a vytvárať množiny vlastných dát
- jednoduché matematické operácie nad dátami
- vynesenie dát do grafu
- výpočet predpokladaného merného tepla na základe zadanej teploty podľa odhadovaných typov príspevkov a ich veľkostí
- fitovanie počítanej krivky k nameraným dátam – teda na základe zmeraných údajov (teplota a merné teplo) určiť veľkosť jednotlivých fyzikálnych príspevkov predpokladaného fyzikálneho modelu
- uloženie rozpracovanej úlohy na disk a jej opätovné načítanie

V tejto práci som sa pokúsil program na základe predložených požiadaviek vytvoriť, pričom súčasťou návrhu bol aj výber vhodných algoritmov nelineárneho fitovania.

Kapitola 2

Fyzikálne pozadie

V teplotnej závislosti merného tepla sa odrážajú prakticky všetky deje v danej látke. Do celkovej hodnoty zistenej experimentom tak prispieva nezávisle niekoľko dejov, napr. merné teplo kmitov kryštálovej mriežky, merné teplo fázových prechodov (supravodiče, magnetické usporiadanie, zmeny kryštálovej štruktúry, ...), merné teplo vodivostných elektrónov a ďalšie. Jednotlivé príspevky sú aditívne a sú obvykle popísané v rámci príslušnej teórie, celková analýza je však pomerne komplikovaná.

Príspevky merného tepla počítané vypracovaným programom sú

- príspevok kmitov mriežky
- merné teplo vodivostných elektrónov
- magnetický príspevok spôsobený postupných obsadzovaním energetických hladín so zvyšujúcou sa teplotou (tzv. Schottkyho príspevok)

Príspevok kmitov kryštálovej mriežky

Tento príspevok, tiež nazývaný fonónový príspevok, je v kryštalických pevných látkach dominantný nad $\sim 10\,\mathrm{K}$. Jeho analýza je preto veľmi dôležitá aj pre vyhodnocovanie ostatných príspevkov. Všeobecne je vibračné spektrum v 3-dimenzionálnej kryštalickej pevnej látke tvorené 3 akustickými a $3(p-1)$ optickými vetvami, pričom p je počet atómov v primitívnej bunke. Správanie akustických kmitov je pomerne dobre popísané v rámci Debyeovho modelu, kde predpokladáme lineárnu závislosť frekvencie kmitov na veľkosti recipročného vektora. Merné teplo je v rámci tohto modelu dané vzťahom [2]

$$C_p = 9k_B N_A \left(T/\theta_D\right)^3 \int_0^{\theta_D/T} \frac{x^4 \mathrm{e}^x}{\left(\mathrm{e}^x - 1\right)^2}\,\mathrm{d}x$$

k_B je Boltzmannova konštanta, N_A je Avogadrova konštanta, T je teplota a θ_D je tzv. Debeyeho teplota, ktorá je určitou charakteristikou kmitov mriežky a predstavuje jediný variabilný parameter.

Optické kmity sú zas dobre popísané v rámci jednoduchšieho Einstenovho modelu, v ktorom predpokladáme frekvenciu kmitov nezávislú na vlnovom vektore. Merné teplo je v rámci tohto modelu dané vzťahom

$$C_p = k_B N_A \sum_i \left(\theta_{E_i}/T\right)^2 \frac{\mathrm{e}^{\theta_{E_i}/T}}{\left(\mathrm{e}^{\theta_{E_i}/T} - 1\right)^2}$$

θ_E je tzv. Einsteinova teplota, ktorá je opäť určitou charakteristikou kmitov mriežky, konkrétne optických vetiev. Podľa počtu vetiev dostávame príslušný počet variabilných parametrov. Kvôli zjednodušeniu modelu je často výhodné popísať viacej vetiev jediným parametrom θ_E.

Merné teplo vodivostných elektrónov

Merné teplo vodivostných elektrónov je spravidla malé, v niektorých látkach však môže najmä pri nízkych teplotách nadobúdať význam a jeho veľkosť je významnou charakteristikou elektrónových vlastností daného materiálu. Závisí lineárne na teplote s koeficientom úmernosti γ (tzv. Sommerfeldov koeficient) [2]

$$C_{el} = \frac{1}{3}\pi^2 n(E_F) k_B^2 T = \gamma T$$

$n(E_F)$ je hustota elektrónových stavov na Fermiho medzi. Pre väčšinu látok je parameter γ teplotne nezávislý a je ho možné pomerne dobre zistiť analýzou nízkoteplotných dát merného tepla. Problematická môže byť analýza u tzv. ťažkofermiónových zlúčenín, kde γ teplotne závislé je.

Schottkyho príspevok

Tento príspevok sa objavuje pri zlúčeninách obsahujúcich najmä ióny vzácnych zemín. Základný elektrónový stav týchto iónov, tzv. multiplet, je pod vplyvom ostatných iónov v kryštálovom okolí rozštiepený na viacero energetických hladín. Ich počet závisí na type daného iónu a veľkosti rozštiepenia. Prípadná degenerácia jednotlivých energetických hladín závisí ešte aj na symetrii kryštálového okolia. Ak poznáme energetické vzdialenosti medzi jednotlivými hladinami, potom entropia (či merné teplo) je daná postupnou zmenou pravdepodobnosti obsadzovania týchto hladín s premennou teplotou.

Pre merné teplo je možné z termodynamiky odvodiť vzťah

$$C_{Sch} = -\frac{\partial^2 \left(k_B T \ln Z\right)}{\partial T^2} = \frac{k_B}{T^2}\left[\sum_i \frac{E_i^2 \mathrm{e}^{-E_i/T}}{Z} - \left(\sum_i \frac{E_i \mathrm{e}^{-E_i/T}}{Z}\right)^2\right]$$

E_i sú jednotlivé energetické hladiny a Z je štatistická suma

$$Z = \sum_i \mathrm{e}^{-E_i/T}$$

K tomuto príspevku sa viaže aj výpočet energetických hladín na základe znalosti symetrie kryštálovej štruktúry a parametrov charakterizujúcich vplyv kryštálového okolia. Vo fyzike pevných látok je tento vplyv popísaný Hamiltoniánom

$$\widehat{H}_{CF} = \sum_{n,m} A_n^m \langle r^n \rangle \, \theta_n \widehat{O}_n^m = \sum_{n,m} B_n^m \widehat{O}_n^m$$

A_n^m, resp. B_n^m sú parametre kryštálového poľa. Počet nezávislých parametrov B_n^m je daný symetriou štruktúry a pohybuje sa medzi 2 (kubická symetria) až 9 (nízke symetrie, napr. ortorombická). $\widehat{O}_n^m$ sú tabelované operátorové ekvivalenty [1], θ_n sú číselné faktory tabelované pre dané ióny vzácnych zemín.

Veľkosť rozštiepenia energetických hladín môže byť ovplyvnená aj ďalšími vonkajšími vplyvmi, napr. pôsobením vonkajšieho magnetického poľa. To je možné započítať do uvedeného Hamiltoniánu, ktorého výpočet je súčasťou vypracovaného programu.

Kapitola 3

Požadované výpočty

Z predložených požiadaviek vyplynula nutnosť implementovať výpočty, ktoré neposkytuje štandardná knižnica tried `Math .NET Framework`. Komplikovanejšie výpočty zahŕňali integráciu krivky v určenom intervale a najmä fitovanie krivky k dátam, teda hľadanie takých parametrov zvoleného modelu, aby namodelovaná krivka čo možno najviac odpovedala vstupným dátam.

3.1 Integrovanie

Presnou a asi najjednoduchšou metódou integrácie danej funkcie, je výpočet rozdielu jej primitívnej funkcie v zadaných okrajoch integračného intervalu. Tento postup (*Newtonov integrál*) ale predpokladá existenciu primitívnej funkcie a jej nájdenie (analytický výpočet integrálu) nie je vždy jednoduché, či možné.

Existuje ale rad numerických integračných metód. Základným typom sú metódy deliace integračný interval na menšie časti a odhadujúce celkový integrál ako sumu integrálov týchto častí [6].

Nech je interval rozdelený bodmi x_i, pričom platí $x_i = x_0 + ih$, kde h je veľkosť každého podintervalu.

Nasledujúce metódy sa líšia počtom bodov, pomocou ktorých je intergál vypočítavaný.

Lichobežníková metóda

$$\int_{x_0}^{x_1} f(x)\,\mathrm{d}x = h\left[\frac{1}{2}f(x_0) + \frac{1}{2}f(x_1)\right] + O\left(h^3 f''\right)$$

f'' je druhá derivácia f vo vhodnom mieste intervalu, cez ktorý sa integruje. Táto metóda dáva presný výsledok pre polynómy do 1. stupňa. Odhad je možné v grafe interpretovať ako výpočet obsahu plochy pod spojnicou funkčných hodnôt krajných bodov intervalu – výpočet obsahu lichobežníka. Odhad je teda počítaný z dvoch hodnôt. Ďalšia metóda využíva hodnoty tri.

Simpsonova metóda

$$\int_{x_0}^{x_2} f(x)\,\mathrm{d}x = h\left[\frac{1}{3}f(x_0) + \frac{4}{3}f(x_1) + \frac{1}{3}f(x_2)\right] + O\left(h^5 f^{(4)}\right)$$

$f^{(4)}$ je 4. derivácia f vo vhodnom mieste intervalu. Počítaná je plocha pod parabolou. Táto metóda dáva vďaka vyrušeniu koeficientov (ľavo-pravá symetria) presný výsledok pre polynómy až do 3. stupňa.

Podobne päťbodová metóda (tzv. *Bodeho*) je presná pre polynómy do 5. stupňa.

Rozšírená lichobežníková metóda

Ak lichobežníkovú metódu aplikujeme na jednotlivé intervaly

$$\langle x_0, x_1\rangle \text{ až } \langle x_{N-1}, x_N\rangle$$

a výsledky sčítame, dostaneme tzv. rozšírené lichobežníkové pravidlo

$$\int_{x_0}^{x_N} f(x)\,\mathrm{d}x = h\left[\frac{1}{2}f(x_0) + \sum_{i=1}^{N-1} f(x_i) + \frac{1}{2}f(x_N)\right] + O\left(\frac{(x_N - x_0)^3 f''}{N^2}\right)$$

Odhad chyby je v takomto zápise závislý na N a nie na h, čo je pre výpočty pohodlnejšie – vo zvolenom intervale $\langle x_0, x_N\rangle$ sa z chyby, ktorú maximálne povoľujeme dosiahnuť, určí počet deliacich bodov a h sa dopočíta.

Rombergova metóda

Táto metóda opakovane aplikuje tzv. *Richardsonovu extrapoláciu* na pravidlo rozšírenej lichobežníkovej metódy. Richardsonova extrapolácia je uskutočňovanie zvoleného výpočtu pre rôzne parametre h a následná extrapolácia výsledkov pre $h = 0$. Algoritmus končí, ak je odhadovaná chyba extrapolácie menšia než požadovaná hodnota.

```
h[0] ← 1
for i ← 1 to steps do
  s[i − 1] ← trapez(function, a, b, i)
  if i ≥ points then
    for j ← 0 to points − 1 do
      h_t[j] ← h[i − steps + j]
      s_t[j] ← s[i − steps + j]
    end for
    extrapol(h_t, s_t, 0, out ss, out dss)
    if |dss| ≤ EPS ∗ |ss| then
      return ss
    end if
```

end if
$h[i] \leftarrow 0{,}25 * h[i-1]$
end for

function je integrovaná funkcia, *steps* je maximálny počet iterácií, *points* je počet bodov v extrapolácii (napr. pre $points = 2$ je výsledkom Simpsonova metóda), *EPS* je požadovaná čiastočná presnosť. Press a kol. [6] volia $points = 5$ a $EPS = 10^{-10}$.

Funkcia `trapez` počíta integrál rozšírenou lichobežníkovou metódou a to pre zadanú funkciu, interval a počet delení. Procedúra `extrapol` polynomicky extrapoluje funkciu $S(h) = \int_{x_0}^{x_N} f(x)\,\mathrm{d}x$ v bode $h = 0$. Pokles h je zámerne volený dvojnásobne oproti skutočnému zvyšovaniu počtu deliacich bodov, vďaka čomu je extrapolácia polynómom nielen v h ale aj v h^2 [6]. `extrapol` vracia extrapoláciu v danom bode a jej chybu.

Výpočet integrálu v Debyeovom modeli je vo vypracovanej aplikácii vykonávaný práve pomocou tejto metódy.

3.2 Lineárna regresia

Najpoužívanejšou metódou regresnej analýzy je *metóda najmenších štvorcov*. Miera súhlasu modelu s dátami je vyjadrená sumou štvorcov odchýlok modelu od dát. Model s najmenšou sumou štvorcov odchýlok je hľadaný model, odtiaľ metóda najmenších štvorcov. Metóda je aplikovateľná na modely, ktoré sú lineárne v hľadaných parametroch.

Špeciálnym prípadom, aplikovaným vo vytvorenom programe, je aproximácia zadaných hodnôt polynómom stupňa 1 – priamkou.

Pre množinu dát s n prvkami (x_i, y_i) teda hľadáme model tvaru

$$y(x) = ax + b$$

Ohodnocovacia funkcia má tvar

$$\chi^2(a,b) = \sum_{i=1}^{n} \left(\frac{y_i - ax_i - b}{\sigma_i} \right)^2$$

kde σ_i je chyba určenia hodnoty y_i [6].

Keďže parametre hľadaného modelu sú súradnice minima χ^2, postačí toto minimum určiť položením derivácií χ^2 rovným nule

$$0 = \frac{\partial \chi^2}{\partial a} = -2 \sum_{i=1}^{n} \frac{x_i \left(y_i - ax_i - b \right)}{\sigma_i^2}$$

$$0 = \frac{\partial \chi^2}{\partial b} = -2 \sum_{i=1}^{n} \frac{y_i - ax_i - b}{\sigma_i^2}$$

Týmto spôsobom je v programe na základe vložených dát určovaný Sommerfeldov koeficient γ.

3.3 Nelineárne regresné modely

Nie všetky modely sú ale lineárnou kombináciou hľadaných parametrov (lineárne), či sa ani nedajú na lineárne previesť – linearizovať. Vzhľadom k variabilite regresných modelov, kritérií regresie a dát, sa však nedajú nájsť všeobecné optimálne algoritmy, ktoré konvergujú ku globálnemu extrému dostatočne rýchlo. Preto sa špeciálne pre účely nelineárnej regresie osvedčuje využívať rôzne výpočtové metódy postupne, či ich kombinovať [3].

Algoritmy nelineárnej regresie sa delia na niekoľko základných skupín

- nederivačné optimalizačné metódy
- derivačné metódy pre kritérium metódy najmenších štvorcov
- všeobecné derivačné metódy
- algoritmy pre špeciálne postupy

V programe sú použité dve najznámejšie metódy, každá z jednej z prvých dvoch skupín.

Simplexové metódy

Algoritmy tohto typu patria medzi nederivačné optimalizačné metódy, ktoré sú v praxi obľúbené najmä vďaka svojej jednoduchosti. Všeobecne sa algoritmy z tejto skupiny odporúča používať vtedy, ak nie je kriteriálna podmienka regresie diferencovateľná alebo nie je diferencovateľná regresná funkcia.

Simplexové metódy vytvárajú adaptívne polyédre v priestore kombinácií možných parametrov modelu. Algoritmus [4] pozostáva z troch fáz

1. konštrukcia prvotného simplexu
2. iteratívny postup k minimu
3. identifikácia ukončenia hľadania

Konštrukcia prvotného simplexu

Súradnice vrcholov polyédra (simplexu) môžeme zapísať do matice s rozmermi $(m+1) \times m$, kde m je počet hľadaných parametrov, pričom každý riadok obsahuje súradnice jedného vrcholu. Ak na začiatku zvolíme pravidelný simplex s dĺžkou hrany t a jedným z vrcholov v počiatku súradníc, matica bude vyzerať takto

$$\mathbf{P} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ c & a & a & \cdots & a \\ a & c & a & \cdots & a \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a & a & a & \cdots & c \end{bmatrix}$$

$$a = \frac{t\left(\sqrt{m+1}-1\right)}{m\sqrt{2}} \qquad c = \frac{t\,(m-1)+\sqrt{m+1}}{m\sqrt{2}}$$

V prípade zadaných počiatočných parametrov je matica konštruovaná tak, že jej prvý riadok obsahuje tento počiatočný odhad. Prvky riadka i sa vypočítajú na základe

$$P_{ij} = \beta_j^{(0)} + \frac{\beta_j^{(0)}}{m\,2\sqrt{2}}\left(m-1+\sqrt{m+1}\right), \qquad i \neq j$$

$$P_{ij} = \beta_j^{(0)} + \frac{\beta_j^{(0)}}{m\,2\sqrt{2}}\left(\sqrt{m+1}-1\right), \qquad i = j$$

$\vec{\beta}^{(0)}$ je vektor parametrov počiatočného odhadu [3], čo je implementované aj vo vypracovanej aplikácii. $\vec{\beta}^{(0)}$ je ale možné umiestniť aj do ťažiska počiatočného simplexu, či simplex natočiť do smeru najväčšieho gradientu ohodnocovacej funkcie.

Iteračný postup

Každý krok iteračného hľadania vyberá medzi troma základnými operáciami: *reflexia*, *kontrakcia* a *expanzia*.

Reflexia sa vykonáva na začiatku každej iterácie. Vychádza z predpokladu, že smer k minimu leží na spojnici vrcholu najhoršieho odhadu ($\vec{P}_h$) a ťažiska simplexu ostatných vrcholov ($\vec{C}$). Pre nový odhad $\vec{P}^*$ získaný reflexiou a vrcholy pôvodného simplexu $\vec{P}_i$ platí

$$\vec{P}^* = \vec{C} + \alpha\left(\vec{C} - \vec{P}_h\right) \qquad \alpha > 0$$

$$\vec{C} = \sum_{i\neq h}^{m} \frac{\vec{P}_i}{m}$$

α je reflexný koeficient. Pre $\alpha = 1$ je nový odhad priemetom $\vec{P}_h$ cez $\vec{C}$.

Označme ohodnocovaciu funkciu ako y a nech $y_i = y\left(\vec{P}_i\right)$. Ak je nový odhad najlepším odhadom, teda $\forall i : y\left(\vec{P}^*\right) < y_i$, tak pokračujeme v hľadaní odhadu vo zvolenom smere expanziou.

$$\vec{P}^{**} = \vec{C} + \gamma\left(\vec{P}^* - \vec{C}\right) \qquad \gamma > 1$$

γ je expanzný koeficient a vyjadruje veľkosť posunu – násobok posunu predtým získaného pri reflexii. Ak je $y\left(\vec{P}^{**}\right) < y\left(\vec{P}^*\right)$, potom $\vec{P}_h \leftarrow \vec{P}^{**}$. Ak nie, tak $\vec{P}_h \leftarrow \vec{P}^*$. Následne pokračujeme ďalšou iteráciou.

Ak $\vec{P}^*$ najlepším odhadom nie je, ale existuje aspoň nejaký horší odhad (okrem $\vec{P}_h$), čiže $\exists\,(i \neq h) : y\Big(\vec{P}^*\Big) < y_i$, potom tiež položíme $\vec{P}_h \leftarrow \vec{P}^*$ a ukončíme prebiehajúcu iteráciu.

Poslednou možnosťou je, že nový odhad je najhorším odhadom. Ak je ale aspoň zlepšením $\vec{P}_h$, potom ešte pred uskutočnením kontrakcie priradíme $\vec{P}_h \leftarrow \vec{P}^*$.

$$\vec{P}^{**} = \vec{C} + \beta\Big(\vec{P}_h - \vec{C}\Big) \qquad 0 < \beta < 1$$

β je kontrakčný koeficient. Nový odhad teda hľadáme v opačnom smere - približovaním k $\vec{C}$. Vzdialenosť $\vec{P}^{**}$ od $\vec{C}$ bude β-násobkom vzdialenosti $\vec{P}_h$ od $\vec{C}$. Ak sme kontrakciou najhorší odhad zlepšili, tak $\vec{P}_h \leftarrow \vec{P}^{**}$ a pokračujeme ďalšou iteráciou. V prípade neúspechu celý simplex pred ďalšou interáciou zmenšíme a to tak, že vzdialenosť každého vrcholu od vrcholu najlepšieho odhadu ($\vec{P}_l$) zmenšíme na λ-násobok

$$\vec{P}_i \leftarrow \vec{P}_l + \lambda\Big(\vec{P}_i - \vec{P}_l\Big) \qquad 0 < \lambda < 1$$

Meloun a Militký [3] odporúčajú voliť koeficienty nasledovne: $\alpha = 1$, $\beta = 0{,}55$, $\gamma = 2{,}9$ a $\lambda = 0{,}5$. Rovnaké koeficienty sú použité aj vo vyhotovenom programe.

Ukončenie hľadania

Jedným z používaných testov ukončenia algoritmu je $|y_h - y_l| < \epsilon$. Možné je ale aj porovnanie [4]

$$\sqrt{\sum_i \frac{\Big(y_i - y\Big(\vec{C}\Big)\Big)^2}{m+1}} < \epsilon$$

Odporúča sa voliť $\epsilon = 10^{-4}$ v prvom, resp. $\epsilon = 10^{-8}$ v druhom prípade [3].

Derivačné metódy

Derivačné metódy pre kritérium najmenších štvorcov patria medzi najpoužívanejšie [3]. Tieto metódy sú použiteľné pre všetky aspoň dvakrát diferencovateľné modelové funkcie. Ich nevýhodou však je lokálna konvergencia, ktorá závisí na voľbe počiatočného odhadu.

Jedná sa o iteračné algoritmy, kedy sa v i-tej iterácii vychádza z odhadu parametrov $\vec{\beta}^{(i)}$, ku ktorému sa priratúva vhodný prírastok $\vec{\Delta}_i$

$$\vec{\beta}^{(i+1)} = \vec{\beta}^{(i)} + \vec{\Delta}_i$$

Postup sa delí na štyri fázy [3]

1. stanovenie prvých odhadov $\vec{\beta}^{(0)}$
2. nájdenie vhodného smerového vektora $\vec{V}_i$

3. určenie α_i tak, aby bol prírastok $\vec{\Delta}_i = \alpha_i \vec{V}_i$ prijateľný

4. test dosiahnutia minima

Stanovenie prvotných odhadov

Stanovenie odhadov je rozhodujúce pre celý proces. Z veľmi zlých odhadov nenájde minimum žiadna z metód tejto skupiny, príp. sa jedná len o lokálne minimum. Naopak z dobrých odhadov väčšinou konvergujú aj menej náročné algoritmy.

Ak sú regresné modely linearizovateľné, je možné počiatočné parametre určiť ich linearizáciou a následnou aplikáciu lineárnej metódy najmenších štvorcov.

Pri interaktívnej práci s počítačom je zas možné ľahko porovnať priebeh modelovej funkcie $f\left(x, \vec{\beta}^{(0)}\right)$ so zadanými dátami a overovať tak kvalitu prvého odhadu [3], na čo sa spolieha aj metóda implementovaná v programe.

Nájdenie smerového vektora

Majme množinu dát (x_i, y_i) a ohodnocovaciu funkciu

$$U\left(\vec{\beta}\right) = \left\|\vec{y} - \vec{f}\right\|^2$$

$\vec{y} = (y_1, \ldots, y_n)^T$, $\vec{f} = \left(f\left(x_1, \vec{\beta}\right), \ldots, f\left(x_n, \vec{\beta}\right)\right)$ a $\|\vec{v}\| = \sqrt{\vec{v}^T \vec{v}}$ je euklidovská norma.

Určíme deriváciu U v mieste $\vec{\Delta} = \vec{\beta} + \alpha \vec{V}$ podľa α.

$$\frac{\partial U\left(\vec{\beta}\right)}{\partial \alpha} = \left(\frac{\partial U\left(\vec{\beta}\right)}{\partial \vec{\beta}}\right)^T \frac{\partial \vec{\beta}}{\partial \alpha} = \vec{g}^T\left(\vec{\beta}\right)\vec{V}$$

pričom gradient $\vec{g}$ je uvažovaný v mieste $\vec{\Delta}$. Pre $\alpha \to 0$ získame smerovú deriváciu S_D

$$S_D = \left.\frac{\partial U\left(\vec{\beta}\right)}{\partial \alpha}\right|_{\alpha \to 0} = \vec{g}^T \vec{V}$$

$$\vec{g} = -2\,\mathbf{J}^T \vec{e} \qquad J_{ij} = \frac{\partial f\left(x_i, \vec{\beta}\right)}{\partial \beta_j} \qquad e_i = y_i - f\left(x_i, \vec{\beta}\right)$$

$\mathbf{J}$ je Jakobiho matica, $\vec{e}$ je vektor diferencií.

Ohodnocovacia funkcia klesá najpríkrejšie v smere $-\vec{g}$. Ak má byť smerový vektor $\vec{V}$ prijateľný (v smere poklesu ohodnocovacej funkcie), tak musí existovať nejaká pozitívne definitná matica $\mathbf{R}$ taká, že

$$\vec{V} = -\mathbf{R}\vec{g}$$

Určenie veľkosti prírastku vektora

Optimálna veľkosť prírastku zvoleného smeru sa určí z Taylorovho rozvoja druhého stupňa funkcie $U\left(\vec{\beta}\right)$

$$U\left(\vec{\beta}+\alpha\vec{V}\right)=U\left(\vec{\beta}\right)+\alpha\vec{g}^{\,T}\vec{V}+\frac{\alpha^2}{2}\vec{V}^T\mathbf{H}\vec{V}$$

$$\mathbf{H}=2\left[\mathbf{J}^T\mathbf{J}+\mathbf{B}\right]\qquad B_{j,k}=\sum_{i=1}^{n}e_i\frac{\partial^2 f\left(x_i,\vec{\beta}\right)}{\partial\beta_j\,\partial\beta_k}$$

H je Hessova matica. Tento Taylorov rozvoj je vzhľadom k α približne kvadratický, čiže optimálne α sa dá stanoviť položením derivácie $U\left(\vec{\beta}+\alpha\vec{V}\right)$ podľa α rovnej 0. Dostaneme

$$\alpha^*=\frac{-\frac{\partial U\left(\vec{\beta}\right)}{\partial\alpha}}{\frac{\partial^2 U\left(\vec{\beta}\right)}{\partial\alpha^2}}=-\vec{g}^{\,T}\vec{V}\left(\vec{V}^T\mathbf{H}\vec{V}\right)^{-1}$$

$$\alpha^*=\vec{g}^{\,T}\mathbf{R}\vec{g}\left(\vec{g}^{\,T}\mathbf{R}^T\mathbf{H}\mathbf{R}\vec{g}\right)^{-1}$$

α^* je tzv. *Raleighov koeficient* a jeho použiteľnosť je obmedzená na oblasť aproximácie uvedeným Taylorovým rozvojom 2. stupňa.

Vektor $\vec{\Delta}_i$, spočítaný na konci každej iterácie, sa obyčajne považuje za prijateľný, ak

$$U\left(\vec{\beta}^{(i)}+\vec{\Delta}_i\right)<U\left(\vec{\beta}^{(i)}\right)$$

Test dosiahnutia optima

Prirodzeným kritériom pre optimum je nulový gradient $\vec{g}$. Rozšírenou terminačnou podmienkou teda je

$$\epsilon>\|\vec{g}\|^2=\sum_{i=1}^{m}{g_i}^2$$

Ďalšou možnosťou je algoritmus ukončiť ak dochádza k príliš malým zmenám veľkosti odhadu.

Marquardtova metóda

Marquardtova metóda patrí medzi tzv. *hybridné metódy*. Kombinuje totiž postup *gradientných* a *newtonových* metód.

Gradientné metódy volia smerový vektor rovný najväčšiemu spádu $-\vec{g}$, teda maticu **R** volia ako jednotkovú. Pre α^* dostaneme

$$\alpha^*=\vec{g}^{\,T}\vec{g}\left[\vec{g}^{\,T}\mathbf{H}\vec{g}\right]^{-1}=\vec{g}^{\,T}\vec{g}\left[\vec{g}^{\,T}\left(\mathbf{J}^T\mathbf{J}\right)^{-1}\vec{g}\right]^{-1}$$

Aj ďaleko od optima sú tieto metódy schopné nájsť smer k minimu. Bohužiaľ ale v blízkosti optima konvergujú pomaly.

Newtonove metódy volia už uvádzaný Taylorov rozvoj pre $\alpha = 1$. Optimálny smerový vektor i-tej iterácie dostaneme pri $\partial U\left(\vec{\beta} + \vec{V}\right) / \partial \vec{V} = \vec{0}$ ako

$$\vec{V_i} = -\mathbf{H}^{-1}\vec{g} = \left(\mathbf{J}^T\mathbf{J} + \mathbf{B}\right)^{-1}\mathbf{J}^T\vec{e}$$

Ak bude U kvadratickou funkciou – eliptický paraboloid, metóda nájde optimum v jedinom kroku. Pre iné funkcie alebo pre vzdialené odhady však konverguje pomaly a navyše vyžaduje znalosť matice druhých derivácií (určenie matice $\mathbf{B}$).

Marquardtova metóda volí smerový vektor ako

$$\vec{V_i}(\lambda) = \left(\mathbf{J}^T\mathbf{J} + \lambda\mathbf{D}_i{}^T\mathbf{D}_i\right)^{-1}\mathbf{J}^T\vec{e}$$

$\mathbf{D}_i$ je diagonálna matica eliminujúca rôzne veľkosti zložiek $\mathbf{J}$ – obyčajne s prvkami na diagonále rovnými diagonálnym prvkom $\mathbf{J}^T\mathbf{J}$. λ je parameter metódy. Jeho vhodná voľba zaisťuje

- pozitívnu definitnosť $\mathbf{R} = \mathbf{J}^T\mathbf{J} + \lambda\mathbf{D}^T\mathbf{D}$, ktorá zaručí, že existuje $\mathbf{R}^{-1}$
- skracovanie vektora $\vec{V_i}$ pri vzďaľovaní sa od optima
- možnosť výberu medzi aproximáciou kvadratúrou a gradientným smerom (aj keď obmedzenú, keďže s väčšou vzdialenosťou – a teda aj smerom približujúcim sa $-\vec{g}$, sa dĺžka $\vec{V_i}$ znižuje)
- obmedzenie dĺžky vektora $\vec{V_i}$ na istú oblasť v okolí $\vec{\beta}^{(i)}$, čo umožňuje zanedbať maticu druhých derivácií $\mathbf{B}$ Newtonovej metódy, čo je ekvivalentné linearizácii regresného modelu

Nevýhodami sú nutnosť pri každej iterácii počítať $\mathbf{R}^{-1}$, pričom pri veľkých λ sú $\vec{V_i}$ príliš malé.

Pôvodný algoritmus pred každou ďalšou iteráciou pri zmenšení U upraví parameter $\lambda \leftarrow \lambda/10$, pričom sa na lepší – teda s menším súčtom štvorcov chýb, upraví aj odhad $\vec{\beta}^{(i)}$. Naopak pri zhoršení U, resp. nevýraznom zlepšení, sa volí $\lambda \leftarrow 10\lambda$, pričom staré parametre ostávajú v platnosti.

Press a kol. [6] odporúčajú začať s $\lambda = 10^{-3}$ a zvýšiť čítač zlých krokov pri každom zhoršení odhadu, resp. nedostatočnom zlepšení. Konkrétne ak

$$\left|U\left(\vec{\beta}^{(i)} + \vec{\Delta}_i\right) - U\left(\vec{\beta}^{(i)}\right)\right| < \epsilon \qquad \epsilon = 0{,}1$$

V prípade zlepšenia sa čítač vynuluje. Ak nasledujú 4 málo úspešné kroky po sebe, tak algoritmus končí [5].

Kapitola 4

Implementácia

4.1 Požiadavky kladené na program

Požadovaná bola grafická aplikácia umožňujúca pohodlnú prácu s dátami získanými z experimentálnej aparatúry. Práca s dátami mala zahŕňať ich úpravu, doplňanie, mazanie. Nad dátami malo byť možné vykonávať jednoduché matematické operácie, ako aj vyniesť ich do grafu.

Ďalšou požiadavkou bol výpočet merného tepla zo zadaného priebehu teplôt podľa zvolených typov a veľkostí jednotlivých príspevkov zvoleného fyzikálneho modelu.

Implementovaná mala byť aj opačná funkcionalita – teda na základe zisteného priebehu teplôt a merného tepla spočítať parametre zvoleného fyzikálneho modelu (fitovať modelovanú krivku na dáta).

Aplikácia mala umožňovať aj uloženie a opätovné načítanie rozpracovanej činnosti.

4.2 Použité nástroje

Kvôli ľahšej prenositeľnosti a aj vďaka predchádzajúcim skúsenostiam, som sa rozhodol napísať požadovaný program v jazyku `C#`.

Vývojové prostredie som zvolil aktuálne a obľúbené v čase zadania práce – Microsoft Visual Studio 2005 pre platformu `Microsoft .NET Framework 2.0`.

4.3 Užívateľské rozhranie

Základným konceptom sa nakoniec stal program poskytujúci dve základné funkcie

- jednoduchý tabuľkový procesor
- zobrazovanie zvolených údajov v grafe

Tieto funkcie sú implementované ako dve záložky hlavného okna.

Tabuľkový procesor

Väčšinu plochy záložky `Data` zaberá miesto pre tabuľku s dátami. Táto tabuľka je implementovná pomocou triedy `Controls.CalcGridView`, čo je trieda odvodená od systémovej triedy `UserControl`.

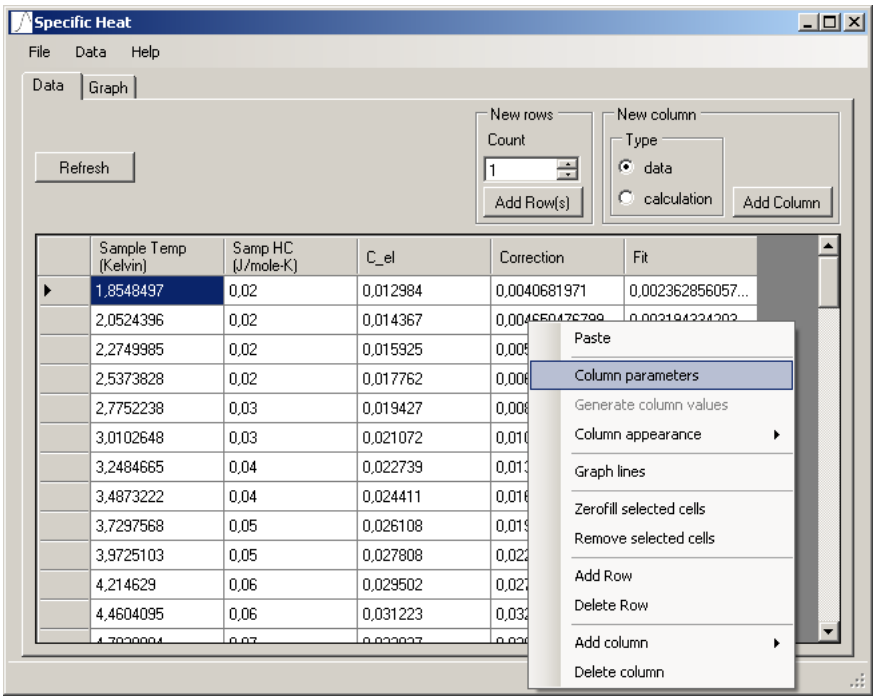
Objekt triedy `CalcGridView` obsahuje inštanciu systémovej triedy `DataGridView` doplnenú o kontextové menu. Zvonku je možné pristupovať k bunkám vnútornej tabuľky (`DataGridViewCell`) pomocou operátora `[]` (hranaté zátvorky), príp. explicitne celý objekt pretypovať na objekt triedy `DataGridView`.

Kontextové menu poskytuje užívateľovi grafické rozhranie, cez ktoré môže v tabuľke manipulovať so stĺpcami a riadkami. Toto rozhranie povoľuje len časť funkčnosti bohatého rozhrania poskytovaného triedou `DataGridView`, resp. toto rozhranie vhodne upravuje.

Stĺpce v tabuľke môžu byť dvojakého druhu. Buď sa jedná o dátové stĺpce (*data*) alebo výpočtové (*calculation*).

Dátové stĺpce sú editovateľné a program nemanipuluje s údajmi, ktoré obsahujú. Údaje vo výpočtových stĺpcoch naopak nie sú priamo editovateľné. Užívateľ len špecifikuje druh výpočtu, jeho parametre, ako aj stĺpec, nad ktorým sa má výpočet uskutočniť.

Výpočet je možné vybrať z predvolených alebo ho špecifikovať zápisom do textového poľa. Výpočet definovaný užívateľom je kombináciou jednoduchých matematických operácií, stĺpcov tabuľky a čísel.



Obr. 4.1: Tabuľkový procesor

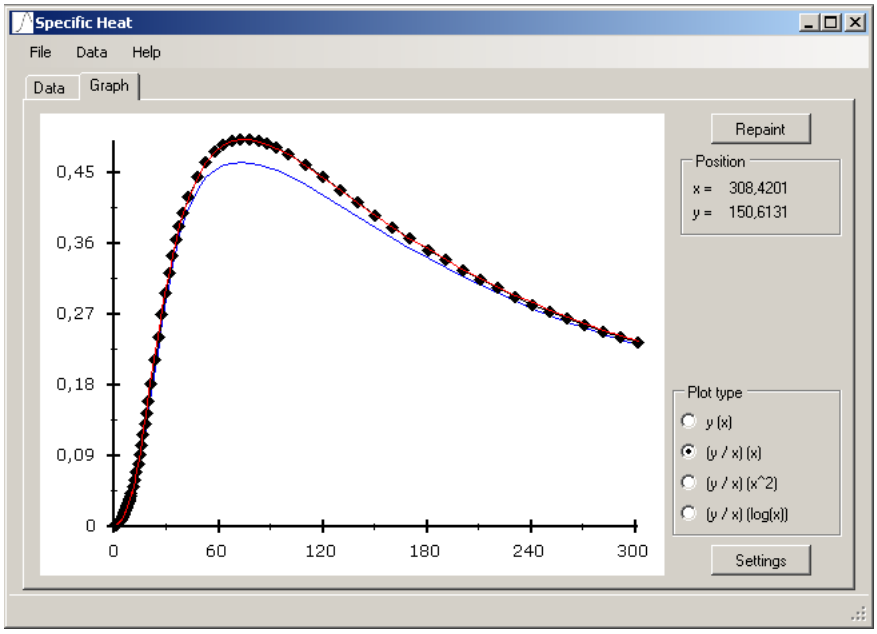
Nastavenie stĺpca je uložené v štruktúre typu `ColumnSettings`. Každý stĺpec má štruktúru so svojím nastavením uloženú v property `Tag`. (*Property* možno preložiť

ako *vlastnosť*, ale radšej ostaneme pri originálnom termíne.)

Graf

Ďalšia záložka zobrazuje vybrané dáta v grafe.

Graf je kreslený na bielu plochu pokrývajúcu takmer celú záložku. Táto plocha prislúcha k objektu systémovej triedy `Panel`. S grafom manipuluje objekt triedy `Graph`, v ktorom sú uložené aj nastavenia pre vykresľovanie.



Obr. 4.2: Graf

Menu

Ostatné užívateľské rozhranie je umiestnené v hlavnom menu aplikácie.

4.4 Dátové štruktúry a ich funkcia

Štruktúra ColumnSettings

Táto štruktúra obsahuje nastavenie jedného stĺpca tabuľky.

Skladá sa z podštruktúry `Fitting`, ďalej atribútu typu `string`, ktorý obsahuje užívateľom definovaný matematický výraz, a objektu triedy `CommandNode`. V objekte `CommandNode` sa užívateľský výraz nachádza rozparsovaný, resp. je v ňom umiestnený koreňový uzol tohto výrazu. Štruktúra obsahuje aj odkaz na stĺpec tabuľky (`DataGridViewColumn`), kde sa nachádzajú zdrojové dáta potrebné pre výpočet.

Štruktúra ColumnSettings.Fitting

Obsahuje atribút vymenovaného typu `CalcType` označujúci druh výpočtu, ďalej samostatný atribút pre každý parameter preddefinovaných výpočtov a pre každý druh výpočtu metódy, ktoré prevádzajú atribúty zodpovedajúce parametrom tohto výpočtu do poľa `double` (výpočtové metódy očakávajú vstupné parametre práve v poli).

Trieda Graph

Sú v nej obsiahnuté nastavenia zobrazovaného grafu (objekt triedy `Graph.Settings`) a metódy pre jeho vykreslenie.

Trieda Graph.Settings

Táto trieda obsahuje pre každú z dvoch osí grafu objekt s nastaveniami osi (podtrieda `AxisSettings` triedy `Graph`) a nastavenia vykresľovaných skupín bodov (príp. kriviek, ak sú vykresľované priebehy funkcie/modelu). Nastavenia skupín bodov sú uložené ako zoznam objektov triedy `Graph.GraphLineSettings`.

V atribúte vymenovaného typu `Graph.PlotType` je uložený spôsob zobrazenia osí grafu. Je možné napr. zobraziť os X v logaritmickej škále a miesto hodnôt y vykresľovať hodnoty pomeru y/x.

Vo vykresľovacích metódach triedy `Graph` sa musia zobrazované hodnoty prepočítať podľa zvoleného `PlotType`. Nato slúžia v triede `Graph.Settings` metódy `PlottedToOriginal` a `OriginalToPlotted`, ktoré počítajú konverzie oboma smermi (v prípade jednoduchého zobrazenia X voči Y k žiadnemu prepočtu nedochádza).

Metóda `SetAxisRanges` určí rozsah zvolenej osi podľa dát, ktoré majú byť zobrazené, teda tak, aby najkrajnejšie zobrazované body boli na okrajoch oblasti, do ktorej sa kreslí.

V nastaveniach grafu sa špecifikuje počet značiek na osi grafu. Ďalšia metóda (`SetAutoMarks`) určí rozostup medzi značkami na zvolenej osi tak, aby jeho hodnota bola zaokrúhlená na najvýznamnejšiu platnú cifru. Podľa tejto hodnoty upraví aj počiatočný bod osi – aby táto cifra bola v počiatku nulová.

V tejto triede je aj odkaz na dátovú tabuľku `CalcGridView`.

Trieda Graph.AxisSettings

Obsahuje atribúty špecifikujúce štýl vykreslenia osi. Teda odsadenie osi od okrajov vykresľovacej oblasti, rozostup medzi značkami na osi (0 pre žiadne značky) a koľká značka má mať popisok (1 – každá, 2 – každá druhá, ...). Ďalej farbu a hrúbku osi, ako aj odsadenie legendy, jej farbu, či formát zobrazenia čísel.

Značky s popisom a bez popisu majú vlastné nastavenia štýlov pomocou podtriedy `MarkSettings`.

Súčaťou triedy `AxisSettings` sú aj prepínače indikujúce, či má byť automaticky počítaný rozsah osi a rozostup medzi jej značkami.

Trieda Graph.GraphLineSettings

Objekt tejto triedy reprezentuje zobrazovanú skupinu bodov, resp. krivku.

V najjednoduchšom prípade sú do grafu vynesené dva stĺpce dátovej tabuľky. Jeden reprezentuje x-ové súradnice zobrazovaných bodov, druhý ich y-ové súradnice.

Atribúty tejto triedy teda špecifikujú dva stĺpce z dátovej tabuľky, symbol, akým majú byť body v grafe reprezentované (vymenovaný typ `Graph.SymbolTypes`), farbu a veľkosť symbolu, príznak (`bool`), či majú byť body spájané úsečkami, príp. aká má byť hrúbka týchto úsečiek. Ďalším atribútom je príznak, či má byť pri každom bode zobrazený chybový interval, ako aj odkaz na stĺpec z tabuľky dát, ktorý príslušné chyby obsahuje.

Druhou možnosťou je, že k vyneseným bodom majú byť spočítané parametre zvoleného modelu. Teda krivka modelu má čo najviac priľnúť k vyneseným bodom. Parametre modelu aj vykresľovanej krivky obsahuje objekt podtriedy `Fitting`.

Trieda Graph.GraphLineSettings.Fitting

Okrem atribútov charakterizujúcich kreslenú fitovanú krivku (farba, hrúbka) obsahuje aj prepínač, či je vôbec fitovanie požadované.

V štruktúre `NotFixedParameters` sú označené parametre, ktoré môžu byť algoritmom pri hľadaní optimálnych parametrov modelu menené, trieda obsahuje aj atribút štruktúry `ColumnSettings.Fitting`, v ktorom sú uchovávané spočítané parametre modelu.

Metóda `Fit` sa na zadanú skupinu bodov (resp. objekt triedy `GraphLineSettings`) pokúsi aplikovať jeden z algoritmov nelineárnej regresie pri hľadaní optimálnych parametrov zvoleného modelu. Typ algoritmu je špecifikovaný ďalším vstupným parametrom metódy. V metóde sa z dátovej tabuľky extrahujú požadované dáta, samotný algoritmus je ale vykonávaný v metódach triedy `Calc`.

Trieda Calc

V tejto statickej triede sú implementované všetky algoritmy popísané v tretej kapitole.

Pôvodne bola v aplikácii implementovaná len Marquardtova fitovacia metóda. V praxi sa však zistilo, že je pre praktické použitie nedostatočná – príliš často odhady končili v lokálnom minime v tesnej blízkosti prvotných odhadov a zlepšili sa len minimálne, či dokonca vôbec.

Preto je okrem metódy `NonLinearFit`, ktorá implementuje Marquardtov algoritmus, implementovaná aj metóda `SimplexFit`, ktorá sa o zlepšenie odhadov pokúša prostredníctvom simplexovej metódy.

Tieto metódy ale nie sú volané priamo. Verejnými metódami sú metódy `CphFit` a `SchottkyFit`, ktoré po kontrole vstupných parametrov zavolajú na základe *bool* parametra `simplex` buď simplexovú alebo Marquardtovu metódu nad danou modelovou funkciou. Vo výstupnom parametri vracajú aj χ^2 výsledného odhadu parametrov.

V niektorých algoritmoch sú volené prísnejšie konštanty (napr. vyšší počet zlých iterácií požadovaný pre ukončenie algoritmu v Marquardtovej metóde), než aké odporúča literatúra, keďže významnejšie neovplyvnili rýchlosť algoritmov. Nezdá sa však, že by mali významný pozitívny vplyv na výsledky.

Konkrétne neschopnosť Marquardtovej metódy opustiť lokálne minimá, ktoré boli umiestnené tesne vedľa alebo dokonca priamo v počiatočných parametroch, nebola odstránená ani týmto experimentovaním.

Pekná konvergencia k dátam ale bola dosahovaná kombinovaním oboch metód. Ak napríklad simplexová metóda už ďalej nedokázala odhad vylepšiť, pomohlo použiť Marquardtov algoritmus, ktorému sa vylepšenie podarilo a ďalšie zlepšenia mohli byť opäť dosiahnuté opakovaným použitím simplexovej metódy.

Trieda `Calc` obsahuje aj metódy počítajúce zložky Hamiltoniánu podľa vstupných parametrov. Podrobný popis hodnôt pre jednotlivé prípady je možné nájsť v literatúre spomínanej vo fyzikálnom úvode tejto práce, konkrétne [1].

Podrobnejšie nie je rozpisovaný ani postup LU dekompozície použitej pri riešení sústavy rovníc v Marquardovej metóde. Alternatívou by mohla byť známejšia Gaussova eliminácia. Počiatočný výber LU dekompozície bol zvolený na základe jej možného použitia pri výpočte inverzných matíc.

Trieda ReadDataFile

Načíta dáta z textového súboru.

Metóda `DoIt` načíta dáta zo súboru vzniknutého ako výstup experimentálnej aparatúry, teda so špecifickou štruktúrou.

Metóda `Ascii` prečíta akýkoľvek textový súbor, kde sú riadky dát zároveň riadkami súboru.

Výstupom je v oboch prípadoch objekt triedy `DataFile`.

Trieda DataFile

Okrem metadát načítaných zo súboru aparatúry obsahuje aj pole názvov stĺpcov a samotné dáta. Dáta sú uložené po riadkoch v zozname `ArrayList`.

Trieda SessionSerializator

Táto trieda slúži na uloženie a opätovné načítanie rozpracovanej úlohy. Využíva pri tom XML serializáciu potrebných objektov.

Ak chceme serializovať nejaký objekt, všetky properties, ktoré z objektu požadujeme uchovať, musíme označiť vhodnými *attributes* (kvôli oddeleniu od termínu atribút nebudeme tento termín prekladať). Také attributes sú napr. `[XmlElement]`, `[XmlArray]` a pod. Pomocou attribute `[XmlIgnore]` naopak property uchovať zakazujeme. Properties, ktoré nemajú priamočiaru reprezentáciu textového reťazca, nahradíme vhodnými ekvivalentami (objekt reprezentujúci stĺpec nahradíme indexom stĺpca, farbu nahradíme jej číselnou reprezentáciou v notácii ARGB a pod.).

Trieda `SessionSerializator` vytvorí objekt triedy `Data`, do ktorého skopíruje tie dáta o stĺpcoch dátovej tabuľky, ktoré chceme uchovať (meno, objekt s nastavením stĺpca `ColumnSettings`, index zdrojového stĺpca – ak uchovávame výpočtový stĺpec atď.) a nastavenie grafu (objekt `Graph.Settings`).

Takto vytvorený objekt potom pomocou metódy `Serialize` uloží na disk a opačne – metódou `Deserialize` z disku načíta a podľa jeho obsahu naplní objekty aplikácie.

Kapitola 5

Záver

Cieľom tejto bakalárskej práce bolo vytvoriť použiteľný program schopný spracovať experimentálne dáta meraní tepelnej kapacity pokusných vzoriek.

Program je schopný spracovať výstupné súbory z experimentálneho zariadenia, ako aj akýkoľvek rozumne štruktúrovaný textový súbor, príp. dáta načítať zo schránky. Získané dáta je možné následne úpraviť základnými matematickými operáciami a vyniesť do grafu.

Program rovnako dokáže spočítať požadované fyzikálne príspevky ako aj ďalšiu požadovanú operáciu – výpočet Hamiltoniánu. Implementované je aj uloženie a opätovné načítanie stavu programu do štruktúrovaného XML súboru.

Možné je aj fitovanie parametrov vybraného modelu na zvolené dáta. Použité sú dva osvedčené algoritmy rôznych typov – Marquardtova a simplexová metóda. Užívateľ tak má možnosť operatívne meniť nielen parametre fitovaného modelu, ale aj samotné postupy hľadania lepších odhadov parametrov. Úspešnosť odhadu si môže overiť jednak hodnotou χ^2, ako aj vizuálnou kontrolou dát vynesených do grafu.

Program je v testovacej prevádzke. Základné požiadavky kladené naň boli splnené. Verím, že postupným zapracovávaním nových pripomienok by sa z neho mohol stať prakticky používaný nástroj a pravidelný pomocník.

Literatúra

[1] Hutchings M. T.: *Point-charge calculations of energy levels of magnetic ions in crystalline electric fields*, Solid State Physics **16** (1964) 227–273.

[2] Kittel, Ch.: *Úvod do fyziky pevných látek*, Academia, Praha, 1985.

[3] Meloun M., Militký J.: *Statistická analýza experimentálních dat*, Academia, Praha, 2004.

[4] Nelder J. A., Mead R.: *A simplex method for function minimization*, Computer Journal **7** (1965) 308–313.

[5] Press W. H., Teukolsky S. A., Vetterling W. T., Flannery B. P.: *Numerical Recipes Example Book (C++)*, Cambridge University Press, Cambridge, 2002.

[6] Press W. H., Teukolsky S. A., Vetterling W. T., Flannery B. P.: *Numerical Recipes in C++*, Cambridge University Press, Cambridge, 2002.

Dodatok A

Adresárová štruktúra CD

`/` – bakalárska práca vo formáte PDF
`/bin` – spustiteľný program pre platformu win32
`/doc` – užívateľská dokumentácia
`/inst` – inštalačné súbory pre platformu win32
`/src` – zdrojové súbory programu
`/test` – testovacie súbory